An Adaptive Evolutionary Multi-objective Approach Based on Simulated Annealing

H. Li lihui10@mail.xjtu.edu.cn
School of Science, Xi'an Jiaotong University, China
Note: research conducted while at The University of Nottingham, United Kingdom

D. Landa-Silva dario.landasilva@nottingham.ac.uk
ASAP Research Group, School of Computer Science, The University of Nottingham, United Kingdom

Abstract

A multi-objective optimization problem can be solved by decomposing it into one or more single objective subproblems in some multi-objective metaheuristic algorithms. Each subproblem corresponds to one weighted aggregation function. For example, MOEA/D is an evolutionary multi-objective optimization (EMO) algorithm that attempts to optimize multiple subproblems simultaneously by evolving a population of solutions. However, the performance of MOEA/D highly depends on the initial setting and diversity of the weight vectors. In this paper, we present an improved version of MOEA/D, called EMOSA, which incorporates an advanced local search technique (simulated annealing) and adapts the search directions (weight vectors) corresponding to various subproblems. In EMOSA, the weight vector of each subproblem is adaptively modified at the lowest temperature in order to diversify the search towards the unexplored parts of the Pareto-optimal front. Our computational results show that EMOSA outperforms six other well-established multi-objective metaheuristic algorithms on both the (constrained) multi-objective knapsack problem and the (unconstrained) multi-objective traveling salesman problem. Moreover, the effects of the main algorithmic components and parameter sensitivities on the search performance of EMOSA are experimentally investigated.

Keywords

Multi-objective Combinatorial Optimization, Pareto Optimality, Evolutionary Multi-objective Algorithms, Multi-objective Simulated Annealing, Adaptive Weight Vectors

1 Introduction

Many real-world problems can be modelled as combinatorial optimization problems, such as knapsack problem, traveling salesman problem, quadratic assignment problem, flowshop scheduling problem, vehicle routing problem, bin packing problem, and university timetabling problem (Papadimitriou and Steiglitz, 1998). Often, these problems are difficult to tackle due to their huge search space, many local optima, and complex constraints. Many of them are NP-hard, which means that no exact algorithms are known to solve these problems in polynomial computation time. In the last two decades, research on combinatorial optimization problems with multiple objectives has attracted growing interest from researchers (Ehrgott and Gandibleux, 2000;

Landa-Silva et al., 2004; Ehrgott, 2005). Due to possible conflicting objectives, optimal solutions for a multi-objective combinatorial optimization (MOCO) problem represent trade-offs among objectives. Such solutions are known as Pareto-optimal solutions. Since the total number of Pareto-optimal solutions could be very large, many practical multi-objective search methods attempt to find a representative and diverse set of Pareto-optimal solutions so that decision-makers can choose a solution based on their preferences.

A number of metaheuristic algorithms including evolutionary algorithms (EA), simulated annealing (SA), tabu search (TS), memetic algorithms (MA) and others (Blum and Roli, 2003; Glover and Kochenberger, 2003; Burke and Kendall, 2005), have been proposed for solving single objective combinatorial optimization problems. Most metaheuristic algorithms try to find global optimal solutions by both diversifying and intensifying the search. Very naturally, these algorithms have also been extended to solve MOCO problems (Gandibleux et al., 2004). Among them, evolutionary multi-objective optimization (EMO) algorithms (Deb, 2001; Tan et al., 2005; Coello Coello et al., 2007) have received much attention due to their ability to find multiple Pareto-optimal solutions in a single run. Pareto dominance and decomposition (weighted aggregation or scalarization) are two major schemes for fitness assignment in EMO algorithms.

Since the 1990s, EMO algorithms based on Pareto dominance have been widely studied. Amongst the most popular methods are PAES (Knowles and Corne, 2000a), NSGA-II (Deb et al., 2002) and SPEA-II (Zitzler et al., 2002). These algorithms have been applied to many real-world and benchmark continuous multi-objective optimization problems. In contrast, EMO algorithms based on decomposition appear to be more successful on tackling MOCO problems. For example, IMMOGLS (Ishibuchi et al., 2003) was applied to the multi-objective flowshop scheduling problem, while MOGLS (Jaszkiewicz, 2002) and MOEA/D (Zhang and Li, 2007) dealt with the multi-objective knapsack problem. In both IMMOGLS and MOGLS, one subproblem with random weight vectors is considered in each generation or iteration. Instead of optimizing one subproblem each time, MOEA/D optimizes multiple subproblems with fixed but uniform weight vectors in parallel in each generation. In most EMO algorithms based on decomposition, local search or local heuristics can be directly used to improve offspring solutions along a certain search direction towards the Pareto-optimal front. These algorithms are also known as multi-objective memetic algorithms (MOMAs) (Knowles and Corne, 2004).

Multi-objective simulated annealing (MOSA) is another class of promising stochastic search techniques for multi-objective optimization (Suman and Kumar, 2006). Several early MOSA-like algorithms (Serafini, 1992; Czyzak and Jaszkiewicz, 1998; Ulungu et al., 1999) define the acceptance criteria for multi-objective local search by means of decomposition. To approximate the whole Pareto-optimal front, multiple weighted aggregation functions with different settings of weights (search directions) are required. Therefore, one of the main challenging tasks in MOSA-like algorithms is to choose appropriate weights for the independent simulated annealing runs or adaptively tune multiple weights in a single run. More recently, MOSA-like algorithms based on dominance have also attracted some attention from the research community (Smith et al., 2008; Sanghamitra et al., 2008).

In recent years, EMO algorithms and MOSA-like algorithms have been investigated and developed along different research lines. However, less attention has been given to the combination of EMO algorithms and simulated annealing. Recently, we investigated the idea of using simulated annealing within MOEA/D for combinatorial optimization (Li and Landa-Silva, 2008). Our preliminary results were very promising. Following that idea, we now make the following contributions in this paper:

- We propose a hybrid between MOEA/D and simulated annealing, called EMOSA, which employs simulated annealing for the optimization of each subproblem and adapts search directions for diversifying non-dominated solutions. Moreover, new strategies for competition among individuals and for updating the external population are also incorporated in the proposed EMOSA algorithm.
- We compare EMOSA to three MOSA-like algorithms and three MOMA-like algorithms on both the multi-objective knapsack problem and the multi-objective traveling salesman problem. The test instances used in our experiments involve two and three objectives for both problems.
- We also investigate the effects of important algorithmic components in EMOSA, such as the strategies for the competition among individuals, the adaptation of weights, the choice of neighborhood structure, and the use of ϵ-dominance for updating the external population.

The remainder of this paper is organized as follows. Section 2 gives some basic definitions and also outlines traditional methods in multi-objective combinatorial optimization. Section 3 reviews related work, while the description of the proposed EMOSA algorithm is given in Section 4. Section 5 describes the two benchmark MOCO problems used here. Section 6 provides the experimental results of the algorithm comparison, while Section 7 experimentally analyzes the components of EMOSA and parameter sensitivities. The final Section 8 concludes the paper.

2 Multi-objective Optimization

This section gives some basic definitions in multi-objective optimization and outlines two traditional multi-objective methods (weighted sum approach and weighted Tchebycheff approach) from mathematical programming.

2.1 Pareto Optimality

A multi-objective optimization problem (MOP) with m objectives for minimization[1] can be formulated as:

$$\text{minimize} \quad F(x) = (f_1(x), \ldots, f_m(x)) \tag{1}$$

where x is the vector of decision variables in the feasible set X, $F : X \to Y \subset R^m$ is a vector of m objective functions, and Y is the objective space. The MOP in (1) is called a *multi-objective combinatorial optimization* (MOCO) problem if X has a finite number of discrete solutions.

For any two objective vectors $u = (u_1, \ldots, u_m)$ and $v = (v_1, \ldots, v_m)$, u is said to dominate v, denoted by $u \prec v$, if and only if $u_i \leq v_i$ for all $i \in \{1, \ldots, m\}$ and there

[1]For maximization, all objectives can be multiplied by -1 to obtain a minimization MOCO problem.

exists at least one index i satisfying $u_i < v_i$. For any two solutions x and y, x is said to dominate y if $F(x)$ dominates $F(y)$. A solution x^* is said to be Pareto-optimal if no solution in X dominates x^*.

The set of all Pareto-optimal solutions in X is called *Pareto-optimal set*. Correspondingly, the set of objective vectors of solutions in the Pareto-optimal set is called *Pareto-optimal front* (POF). The lower and upper bounds of the POF are called the ideal point z^* and the nadir point z^{nad} respectively, that is, $z_i^* = \min_{u \in \text{POF}} u_i$ and $z_i^{nad} = \max_{u \in \text{POF}} u_i$, $i = 1, \ldots, m$.

A solution x is said to ϵ-dominate y if $F(x) - \epsilon$ dominates $F(y)$, where $\epsilon = (\epsilon_1, \ldots, \epsilon_m)$ with $\epsilon_i > 0, i = 1, \ldots, m$. If x dominates y, x also ϵ-dominates y. However, the reverse is not necessarily true, see (Deb et al., 2005) for more on ϵ-dominance.

2.2 Traditional Multi-objective Methods

In mathematical programming, a multi-objective optimization problem is often converted into one or multiple single objective optimization subproblems by using linear or nonlinear aggregation of objectives with some weights. This is called decomposition or scalarization. In this section, we describe two commonly-used traditional methods - weighted sum approach and weighted Tchebycheff approach (Miettinen, 1999).

- Weighted Sum Approach

  This method considers the following single objective optimization problem:

  $$\text{minimize} \quad g^{(ws)}(x, \lambda) = \textstyle\sum_{i=1}^{m} \lambda_i f_i(x) \tag{2}$$

  where $x \in X$, $\lambda = (\lambda_1, \ldots, \lambda_m)$ is the weight vector with $\lambda_i \in [0, 1], i = 1, \ldots, m$ and $\sum_{i=1}^{m} \lambda_i = 1$. Each component of λ can be regarded as the preference of the associated objective. A solution x^* of the MOP in (1) is a *supported optimal solution* if it is the unique global minimum of the function in (2).

- Weighted Tchebycheff Approach

  The aggregation function of this method has the following form:

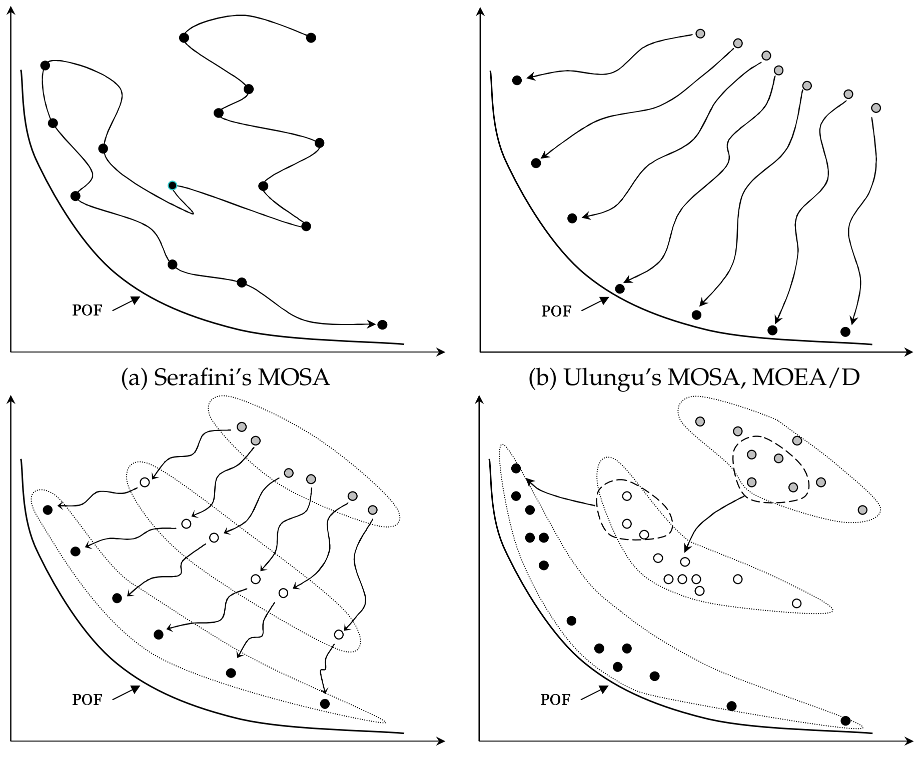
  $$\text{minimize} \quad g^{(tch)}(x, \lambda) = \max_{i \in \{1, \ldots, m\}} \lambda_i |f_i(x) - z_i^*| \tag{3}$$

  where $x \in X$, λ is the same as above, and z^* is the ideal point.

Under some mild conditions, the global minimum of the function in (2) or (3) is also a Pareto-optimal solution of the MOP in (1). To find a diverse set of Pareto-optimal solutions, a number of weight vectors with evenly spread distribution across the trade-off surface should be provided. The weighted Tchebycheff approach has the ability to deal with non-convex POF but the weighted sum approach lacks this ability. Normalization of objectives is often needed when the objectives are incommensurable, i.e. have different scales.

3 Related Previous Work

Extensions of simulated annealing for multi-objective optimization have been studied for about twenty years. Several well-known MOSA-like algorithms developed in the



(a) Serafini's MOSA

(b) Ulungu's MOSA, MOEA/D

(c) Czyzak's MOSA

(d) IMMOGLS and MOGLS

Figure 1: Graphical illustration of different strategies used in multi-objective metaheuristics based on decomposition.

1990s use weighted aggregation for their acceptance criteria in local search. The main differences between the various MOSA-like algorithms lie in the choice of weights during the search. The MOSA proposed by Serafini (1992) is a single-point method that optimizes one weighted aggregation function in each iteration (see Figure 1(a)). In this method, the current weight vector is smoothly changed as the search progresses. Similarly, IMMOGLS and MOGLS optimize a single weighted aggregation function in each iteration (see Figure 1(d)). In contrast, two MOSA-like algorithms (Ulungu et al., 1999; Czyzak and Jaszkiewicz, 1998) are population-based search methods and optimize multiple weighted aggregation functions (see Figure 1(b) and (c)). In Ulungu's MOSA, a set of evenly-spread fixed weights is needed, while Czyzak's MOSA adaptively tunes the weight vector of each individual according to its location with respect to the nearest non-dominated neighbor.

In Zhang and Li (2007), a multi-objective evolutionary algorithm based on decomposition, called MOEA/D, was proposed. It optimizes multiple subproblems simultaneously by evolving a population of individuals. In this algorithm, each individual corresponds to one subproblem for which a weighted aggregation function is defined. Like Ulungu's MOSA, MOEA/D uses fixed weights during the whole search. Since subproblems with similar weight vectors have optimal solutions close to each other, MOEA/D optimizes each subproblem by using the information from the optimization of neighboring subproblems. For this purpose, recombination and competition between neighboring individuals are taken into consideration for mating restriction. Unlike other EMO algorithms, MOEA/D provides a general framework which allows the application of any single objective optimization technique to its subproblems.

In Li and Landa-Silva (2008), we proposed a preliminary approach hybridizing MOEA/D with simulated annealing. In the present work, competition and adaptation of search directions are incorporated to develop an effective hybrid evolutionary approach, called EMOSA. In the proposed algorithm, the current solution of each subproblem is improved by simulated annealing with different temperature levels. After certain low temperature levels, the weight vectors of subproblems are modified in the same way as Czyzak's MOSA. Contrary to the original MOEA/D, no crossover is performed in our hybrid approach. Instead, diversity is promoted by allowing uphill moves following the simulated annealing rationale.

The hybridization of population-based global search coupled with local search is the main idea of many memetic algorithms (also called genetic local search), which have shown considerable success in single objective combinatorial optimization (Merz, 2000; Krasnogor, 2002; Hart et al., 2004). In these algorithms, genetic search is used to explore promising areas of the search space (diversification) while local search is applied to examine high-quality solutions in a specific local area (intensification). There have also been some extensions of memetic algorithms for multi-objective combinatorial optimization (Knowles and Corne, 2004).

In some well-known multi-objective memetic algorithms based on decomposition, such as MOGLS and MOEA/D, two basic strategies (random weights and fixed weights) have been commonly used to maintain diversity of search directions towards the POF. However, both strategies have their disadvantages. On the one hand, random weights might provoke the population to get stuck in local POF or approximate the same parts of the POF repeatedly. On the other hand, fixed weights might not be able to cover the whole POF well. In Czyzak and Jaszkiewicz (1998), the dispersion of non-dominated solutions over the POF is controlled by tuning the weights adaptively.

In our previous work (Li and Landa-Silva, 2008), the weight vector of each individual is modified on the basis of its location with respect to the closest non-dominated neighbor when the temperature goes below certain level. The value of each component in the weight vector is increased or decreased by a small value. However, the diversity of all weight vectors was not considered. The modified weight vector could be very close to others if the change is too big. In this case, the aggregation function with the modified weights might not be able to guide the search towards unexplored parts of the POF. To overcome this weakness, the adaptation of each weight vector should consider the diversity of all weight vectors in the current population.

As shown in Li and Zhang (2009), competition between solutions of neighboring subproblems in MOEA/D could affect the diversity of the population. Particularly, when the population is close to the POF, the competition between neighboring solutions provokes duplicated individuals in the population. Then, in some cases, competition should not be encouraged. For this reason, we need an adaptive mechanism to control competition at different stages during the search.

Like in many multi-objective metaheuristics, MOEA/D uses an external population to store non-dominated solutions. When the POF is very large, maintaining such population is computationally expensive. No diversity strategy is adopted to control the size of the external population in MOEA/D. Popular diversity strategies are crowding distance in NSGA-II and the nearest neighbor method in SPEA-II. Both strategies have computational complexity of at least $\mathcal{O}(L^2)$, where L is the current size of the external population. The ϵ-dominance concept is a commonly-used technique for fitness assignment and diversity maintenance (Grosan, 2004; Deb et al., 2005). We use ϵ-dominance within EMOSA to maintain the diversity of the external population of non-dominated solutions.

4 Description of EMOSA

In this section, we propose EMOSA, a hybrid adaptive evolutionary algorithm that combines MOEA/D and simulated annealing. Like in MOEA/D, multiple single objective subproblems are optimized in parallel. For each subproblem, a simulated annealing based local search is applied to improve the current solution. To diversify the search towards different parts of the POF, the weight vectors of subproblems are adaptively modified during the search. The main features of EMOSA include: 1) maintenance of adaptive weight vectors; 2) competition between individuals with respect to both weighted aggregation and Pareto dominance; 3) use of ϵ-dominance for updating the external population. In this section, we first present the main algorithmic framework of EMOSA and then discuss its algorithmic component in more detail.

4.1 Algorithmic Framework

Procedure 1 EMOSA

```
1: initialize $Q$ weight vectors $\Lambda = \{\lambda^{(1)}, \ldots, \lambda^{(Q)}\}$ by SelectWeightVectors($Q$)
2: generate $Q$ initial individuals POP = $\{x^{(1)}, \ldots, x^{(Q)}\}$ randomly or using heuristics
3: initialize external population EP with non-dominated solutions in POP
4: initialize the temperature setting: $T \leftarrow T_{max}$, $\alpha \leftarrow \alpha_1$
5: repeat
6:   for all $s \in \{1, \ldots, Q\}$ do
7:     calculate the neighborhood $\Lambda(s)$ for the $s$-th weight vector $\lambda^{(s)}$
8:   end for
9:   repeat
10:    estimate the nadir point $z^{nad}$ and the ideal point $z^*$ using EP
11:    for all $s \in \{1, \ldots, Q\}$ do
12:      apply SimulatedAnnealing($x^{(s)}, \lambda^{(s)}, T$) to $x^{(s)}$ and obtain $y$
13:      compete $y$ with the current population POP by UpdatePopulation($y, s$)
14:    end for
15:    decrease the current temperature value by $T \leftarrow T \times \alpha$
16:  until $T < T_{min}$
17:  for all $s \in \{1, \ldots, Q\}$ do
18:    modify the weight vector $\lambda^{(s)}$ by AdaptWeightVectors($s$)
19:  end for
20:  reset the temperature settings: $T \leftarrow T_{reheat}$ and $\alpha \leftarrow \alpha_2$
21: until stopping conditions are satisfied
22: return EP
```

Two sets of weight vectors and two populations of individuals are maintained: 1) the current set of Q weight vectors $\Lambda = \{\lambda^{(1)}, \dots, \lambda^{(Q)}\}$ and the candidate set Θ of weight vectors; 2) the current population POP $= \{x^{(1)}, \dots, x^{(Q)}\}$ and the external population EP. The candidate weight vectors in Θ are generated by using uniform lattice points (see details in Section 4.2). The size of Θ is much larger than Q. The total number of weight vectors in the current weight set Λ equals the population size Q since each individual is associated to one weight vector.

The framework of EMOSA is shown in **Procedure 1**. Note that Θ, Λ, POP, and EP are global memory structures, which can be accessed from any subprocedure of EMOSA. In the following, we explain the main steps in EMOSA.

1. The initial weight vectors and population are generated in lines 1 and 2. The external population EP is formed with the non-dominated solutions in POP (line 3).
2. For each weight vector $\lambda^{(s)}$, the associated neighborhood $\Lambda(s)$ is computed based on the distance between weight vectors in lines 6-8. Once the current temperature level T goes below the final temperature level T_{min}, the neighborhood should be updated.
3. The nadir and ideal points are estimated using all non-dominated solutions in EP (line 10). More precisely, $f_i^{max} = \max_{x \in \text{EP}} f_i(x)$, $f_i^{min} = \min_{x \in \text{EP}} f_i(x)$, $i = 1, \dots, m$. These two points are used in the setting of ϵ-dominance.
4. For each individual in the population, simulated annealing local search is applied for improvement (line 12). The resulting solution y is then used to update other individuals in the population (line 13).
5. If the current temperature T is below the final temperature T_{min}, the weight vector of each individual is adaptively modified (lines 17-19).
6. In line 4, the current temperature level T is set to T_{max} (starting temperature level), and the current cooling rate is set to be α_1 (starting cooling rate). The current temperature is decreased in a geometric manner (line 15). To help the search escape from the local optimum, the current temperature is re-heated (line 20) to T_{reheat} ($< T_{max}$) and a faster annealing scheme is also applied with $\alpha = \alpha_2$ ($< \alpha_1$).

More details of the main steps in EMOSA are given in the following sections.

4.2 Generation of Diverse Weight Vectors

Here, Θ is the set of all normalized weight vectors with components chosen from the set: $\{0, 1/H, \dots, (H-1)/H, 1\}$, with H a positive integer number. The size of Θ is C_{H+m-1}^{m-1}, with m the number of objectives. Figure 2 shows a set of 990 uniformly-distributed weight vectors produced using this method for $H = 43$ and $m = 3$.

In the initialization stage of EMOSA, Q initial weight vectors evenly spread are selected from the candidate weight set Θ. Note that the size of Θ is much larger than Q. This procedure **SelectWeightVectors** is shown in **Procedure 2**. In lines 1-3, the extreme weight vectors are generated. Each of these extreme weight vectors corresponds to the optimization of a single objective. In line 4, c is the number of weight vectors selected so far and all selected weight vectors are copied into a temporary set Φ. In line 6, the set A of all weight vectors in Θ with the same maximal distance to Φ are identified. In

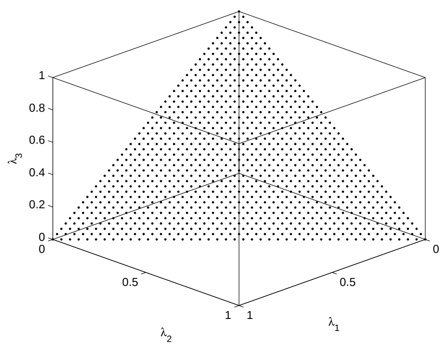


Figure 2: 990 uniform weight vectors.

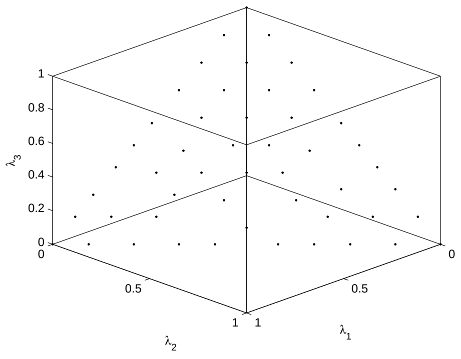


Figure 3: 50 selected weight vectors.

Procedure 2 SelectWeightVectors (Q)

```
1: for all $s \in \{1, \dots, m\}$ do
2:     $\lambda_j^{(s)} \leftarrow 1$ if $j = s$; otherwise $\lambda_j^{(s)} \leftarrow 0, j \in \{1, \dots, m\} \backslash \{s\}$.
3: end for
4: $c \leftarrow m$ and $\Phi = \{\lambda^{(1)}, \dots, \lambda^{(c)}\}$.
5: repeat
6:     $A \leftarrow \{\lambda | \lambda = \text{argmax}_{v \in \Theta} \text{dist}(v, \Phi)\}$. where $\text{dist}(v, \Phi) = \min_{u \in \Phi} \text{dist}(v, u)$.
7:     $B \leftarrow \{\lambda^{(c)}, \lambda^{(c-1)}, \dots, \lambda^{(\lfloor c/2 \rfloor)}\}$
8:     $\lambda^{(c+1)} \leftarrow \text{argmax}_{u \in A} \text{dist}(u, B)$
9:     $\Phi = \Phi \cup \{\lambda^{(c+1)}\}$ and $c \leftarrow c + 1$.
10: until $c = Q$
11: return $\Lambda = \{\lambda^{(1)}, \dots, \lambda^{Q}\}$
```

line 7, the set B is formed with half of the recently selected weight vectors. In line 8, the weight vector in A with the maximal distance to B is chosen as the next weight vector in Λ. In this way, all weight vectors selected so far are well-distributed. For example, Figure 3 shows the distribution of 50 weight vectors selected from the initial 990 weight vectors shown in Figure 2.

The procedure **SelectWeightVectors** involves $\mathcal{O}(Q^2 \times |\Theta|)$ basic operations in lines 6 and 7 as well as $\mathcal{O}(|\Theta|^2)$ distance calculations between the weight vectors in Θ. Ideally, the large size of Θ leads to a good approximation to the POF since each Pareto optimal solution is the optimal solution of a certain scalarizing function. However, as the size of Θ increases, the computational complexity of **SelectWeightVectors** and also **AdaptWeightVectors** (see details in the next subsection) increase. But, a small size of Θ might not be enough to approximate the whole Pareto front. To guarantee efficiency, H should be set properly.

In EMOSA, the neighborhood of each weight vector is determined using the same method as in MOEA/D. More precisely, the neighborhood $\Lambda^{(s)}$ of $\lambda^{(s)}$ for $s = 1, \dots, Q$, is formed by the indexes of its K closest neighbors in Λ. Note that the neighborhoods of weight vectors in MOEA/D remain unchanged during the whole search process while those in EMOSA are adaptively changed. This procedure involves $\mathcal{O}(Q^2)$ distance cal-

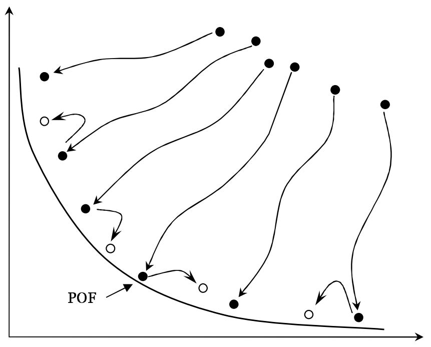


Figure 4: Weight adaptation in EMOSA.

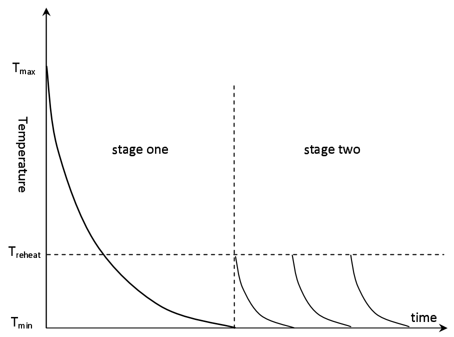


Figure 5: Cooling schemes in EMOSA.

culations and $\mathcal{O}(K \times Q^2)$ basic comparison operators for sorting these distances.

In Czyzak's MOSA, the basic idea is to move each non-dominated solution away from its nearest neighbor. To implement this, the weight vector λ of each individual x is modified at each local move step according to the location of the nearest non-dominated neighbor of x, lets say y. The components of λ are changed as follows:

$$\lambda_i = \begin{cases} \min\{\lambda_i + \delta, 1\} & \text{if } f_i(x) < f_i(y) \\ \max\{\lambda_i - \delta, 0\} & \text{otherwise} \end{cases} \tag{4}$$

where $\delta > 0$ is the variation level. Note that λ should be normalized after the above change. However, there are two weaknesses in this method. First, the setting of δ should be provided empirically. On the one hand, a large value of δ could push the current solution towards the neighborhood of other solutions. On the other hand, a very small value of δ might not be able to guide the search towards unexplored areas of the POF. Second, the diversity of all weight vectors in the current population is not considered in setting the value of λ. This will reduce the chance to approximate different parts of the POF with the same possibility.

Procedure 3 AdaptWeightVectors(s)

```
1: find the closest non-dominated neighbor $x^{(t)}, t \in \{1, \dots, Q\}$ to $x^{(s)}$
2: $A \leftarrow \{\lambda \in \Theta | \text{dist}(\lambda^{(s)}, \lambda^{(t)}) < \text{dist}(\lambda, \lambda^{(t)})$ and $\text{dist}(\lambda, \lambda^{(s)}) \le \text{dist}(\lambda, \Lambda)\}$
3: if A is not empty then
4:     $\lambda^{(s)} \leftarrow \text{argmax}_{u \in A} \text{dist}(u, \lambda^{(s)})$
5: end if
6: return $\lambda^{(s)}$
```

To overcome the above weaknesses, we introduce a new strategy in EMOSA, illustrated in Figure 4, for adapting weight vectors when $T < T_{\min}$, i.e. when the simulated annealing enters the only improving phase. The corresponding procedure **AdaptWeightVectors** is shown in **Procedure 3**. Instead of changing the components of the weight vectors, our strategy picks one weight vector λ from the candidate set Θ to replace the current one $\lambda^{(s)}$. For the current solution $x^{(s)}$, we need to find its closest non-dominated neighbor $x^{(t)}$, which corresponds to $\lambda^{(t)}$. The selected weight vector λ must satisfy two conditions: 1) dist($\lambda^{(s)}, \lambda^{(t)}$) $<$ dist($\lambda, \lambda^{(t)}$), and 2) dist($\lambda, \lambda^{(s)}$) $\leq$ dist(λ, Λ). Condition 1) indicates that the selected weight vector should increase the distance between the weight vectors of $x^{(s)}$ and $x^{(t)}$. This could cause an increase in the distance between $x^{(s)}$ and $x^{(t)}$ in the objective space. Condition 2) guarantees that the selected weight vector should not be too close to other weight vectors in the current population. In line 2 of Procedure 3, all weight vectors in Θ satisfying these two conditions are stored in A. From this set, the weight vector with the maximal distance to $\lambda^{(s)}$ is selected. The procedure **AdaptWeightVectors** involves $\mathcal{O}(Q \times |\Theta|)$ distance calculations when the current temperature is below the value of T_{min}.

4.3 Local Search and Evolutionary Search

In EMOSA, each individual in the population is improved by a local search procedure, namely **SimulatedAnnealing** shown in **Procedure 4**. Then, the improved solutions are used to update other individuals in the neighborhood as shown in **Procedure 5 UpdatePopulation**.

Procedure 4 SimulatedAnnealing($x^{(s)}, \lambda^{(s)}, T$)

```
1: $y \leftarrow x^{(s)}$
2: repeat
3:    generate neighboring solution $x' \in \mathcal{N}(y)$
4:    if $F(x')$ is not dominated by $F(y)$ then
5:       update EP in terms of $\epsilon$-dominance
6:    end if
7:    calculate the acceptance probability $P(y, x', \lambda^{(s)}, T)$
8:    if $P(y, x', \lambda^{(s)}, T) > random[0, 1)$ then
9:       $y \leftarrow x'$
10:   end if
11: until Stopping conditions are satisfied
12: return $y$
```

In line 3 of **Procedure 4**, a neighboring solution x' is generated from $\mathcal{N}(y)$, the neighborhood of the current solution y. If $F(x')$ is not dominated by $F(y)$, then update EP in terms of ϵ-dominance. The components of ϵ are calculated by $\epsilon_i = \beta \times (z_i^{nad} - z_i^*), i = 1, \ldots, m$. Here, $\beta > 0$ is a parameter to control the density of solutions in EP. The update of EP using ϵ-dominance benefits two aspects: 1) maintain the diversity of EP and 2) truncate the solutions of EP in dense areas of the objective space. In this work, β is set to 0.002 for two-objective problems and 0.005 for three-objective problems.

In the simulated annealing component of EMOSA, the probability of accepting neighboring solutions is given by:

$$P(x, x', \lambda, T) = \min\{1, \exp(-\tau \times \frac{g(x', \lambda) - g(x, \lambda)}{T})\} \tag{5}$$

where $T \in (0, 1]$ is the normalized temperature level, τ is a problem-specific balance constant, and g is a weighted aggregation function of x with the weight vector λ, which can be the weighted sum function $g^{(ws)}$ or the weighted Tchebycheff approach $g^{(te)}$ or the combination of both. According to (5), more uphill (worse) moves are accepted with high probability when T is close to 1, but increasingly only downhill (better) moves are accepted as T goes to zero. The acceptance of worse neighboring solutions reduces the possibility of getting trapped in local optima.

The degree of uphill moves acceptance is determined by the annealing schedule, which is crucial to the performance of simulated annealing. In EMOSA, we apply the following two annealing schedules, shown in Figure 5, in two stages.

- Schedule 1: Before the current temperature T goes below the final temperature T_{min} for the first time (early stage), local search seeks to improve all individuals using high temperature T_{max} and slow cooling rate α_1 (line 4 in **Procedure 1**). This is the same as many other simulated annealing algorithms. The main task at this stage is to start from the initial population and then quickly find some representative solutions in the POF from the initial population.
- Schedule 2: Whenever the current temperature T goes below the final temperature T_{min} (late stage), T is increased to a medium temperature level T_{reheat}, and a faster cooling rate $\alpha_2 (< \alpha_1)$ is applied (line 20 in **Procedure 1**). Since all solutions in the current population should by now be close to the POF, they should also be close to the optimal solutions of aggregation functions with modified weight vectors. For this reason, there is no need to start a local search from a high temperature level. Fast annealing could help the intensification of the search near the POF since not many uphill moves are allowed.

Moreover, the parameter τ in (5) should be set empirically. As suggested in Smith et al. (2004), about half of non-improving neighbors should be accepted at the beginning of simulated annealing. Based on this idea, we set τ to $-\log(0.5)T_{max}/\bar{\Delta}$ in EMOSA, where $\bar{\Delta}$ is the average of Δg values in the first 1000 uphill moves. For these first 1000 uphill moves, the acceptance probability is set to 0.5. In EMOSA, the **SimulatedAnnealing** procedure is terminated after examining $\#ls$ neighbors. Hence, at each temperature level, this procedure mainly involves $Q \times \#ls$ function evaluations and the update of the external population EP.

Note that EMOSA optimizes different subproblems using the same temperature cooling scheme. Actually, each subproblem can also be solved under different cooling schemes. This idea has been used in previous work on simulated annealing algorithm for multi-objective optimization (Tekinalp and Karsli, 2007) and also for single objective optimization (Burke et al., 2001).

Procedure 5 UpdatePopulation(y, s)

```
1: if $g(y, \lambda^{(s)}) < g(x^{(s)}, \lambda^{(s)})$ then
2:     $x^{(s)} \leftarrow y$
3: end if
4: for all $k \in \Lambda(s)$ and $k \neq s$ do
5:     if $F(y)$ dominates $F(x^{(k)})$ then
6:         $x^{(k)} \leftarrow y$
7:     end if
8: end for
```

The key difference between EMOSA and other population-based MOSA-like algorithms, such as Ulungu and Czyzak's algorithms, is the competition between individuals. As shown in Li and Zhang (2009), the competition may affect the balance between diversity and convergence in MOEA/D. On the one hand, the competition among individuals could speed up the convergence towards the POF when the current population is distant to the POF. On the other hand, the competition between individuals close to the POF could cause loss of diversity. This is because one solution that is non-dominated with respect to its neighbors could be worse if the comparison is made using a weighted aggregation function. In this paper, we apply different criteria to update the current solution and its neighbors. The details of the process for updating the population are shown in **Procedure 5**. In lines 1-3, the current solution $x^{(s)}$ is compared to the solution y obtained by local search and the comparison is made using the weighted aggregation function. In lines 4-8, the neighbors of $x^{(s)}$ are replaced only if they are dominated by y. By doing this, the selection pressure among individuals is weak when the current population is close to the POF. Therefore, the diversity of individuals in the neighborhood can be preserved.

5 Two Benchmark MOCO Problems

In this paper, we consider two $\mathcal{NP}$-hard MOCO test problems, the multi-objective knapsack problem and the multi-objective traveling salesman problem. These problems have been widely used when testing the performance of various multi-objective metaheuristics (Jaszkiewicz, 2002; Zhang and Li, 2007; Paquete and Stützle, 2003; Garcia-Martinez et al., 2007; Li et al., 2010).

5.1 The 0-1 Multi-objective Knapsack Problem

Given n items and m knapsacks, the 0-1 multi-objective knapsack problem (MOKP) can be formulated as:

$$\text{maximize} \quad f_i(x) = \textstyle\sum_{j=1}^{n} p_{ij} x_j, \tag{6}$$

$$\text{subject to} \quad \textstyle\sum_{j=1}^{n} w_{ij} x_j \le c_i, \;\; i = 1, \ldots, m \tag{7}$$

where $x = (x_1, \ldots, x_n)$ is a binary vector. That is, $x_j = 1$ if item j is selected to be included in the m knapsacks and $x_j = 0$ otherwise, p_{ij} and w_{ij} are the profit and weight of item j in knapsack i, respectively, and c_i is the capacity of knapsack i.

Due to the constraint on the capacity of each knapsack, the solutions generated by local search moves or genetic operators could be infeasible. One way to deal with this is to design a heuristic for repairing infeasible solutions. Some repair heuristics have been proposed in the literature (Zitzler and Thiele, 1999; Zhang and Li, 2007). In this paper, we use the same greedy heuristic adopted in Zhang and Li (2007) for constraint handling. The basic idea is to remove some items with heavy weights and little contribution to an objective function from the overfilled knapsacks.

In Knowles and Corne (2000b), neighboring solutions for the MOKP are produced by flipping each bit in the binary vector with probability k/n, called k-bit flipping here. For example, if the j-th bit of x is chosen for flipping, then $x_j = 1 - x_j$. After flipping, the modified solution x might be infeasible and a repair heuristic might be needed. In this paper, we suggest a new neighborhood structure, k-bit insertion, in which only the bits equal to zero may be flipped. This means that more items are added into the knapsacks but no items are removed. Consequently, the modified solution after flipping bits with value of zero is very likely to be infeasible but will have higher profits. Then, the repair heuristic is called to remove some items as described above until the solution becomes feasible again.

5.2 The Multi-objective Traveling Salesman Problem

The Traveling Salesman Problem (TSP) can be modeled as a graph $G(V, E)$, where $V = \{1, \dots, n\}$ is the set of vertices (cities) and $E = \{e_{i,j}\}_{n \times n}$ is the set of edges (connections between cities). The task is to find a Hamiltonian cycle of minimal length, which visits each vertex exactly once. In the case of the multi-objective TSP (MOTSP), each edge $e_{i,j}$ is associated with multiple values such as cost, length, traveling time, etc. Each of them corresponds to one criterion. Mathematically, the MOTSP can be formulated as:

$$\text{minimize } f_i(\pi) = \sum_{j=1}^{n-1} c^{(i)}_{\pi(j),\pi(j+1)} + c^{(i)}_{\pi(1),\pi(n)} \quad i = 1, \dots, m \tag{8}$$

where $\pi = (\pi(1), \dots, \pi(n))$ is a permutation of cities in Π, the set of all permutations of $I = \{1, \dots, n\}$ and $c^{(i)}_{s,t}, s, t \in I$ is the cost of the edge between city s and city t regarding criterion i. Unlike the MOKP problem, no repair strategy is needed for solutions to the MOTSP problem because all permutations represent feasible solutions.

Like in the single objective TSP problem, the neighborhood move used here for the MOTSP is the 2-opt local search procedure constructed in two steps: 1) remove two non-adjacent edges from the tour and then 2) reconnect vertices associated with the removed edges to produce a different neighbor tour.

6 Computational Experiments

In this section, we experimentally compare the proposed EMOSA to three other MOSA-like algorithms, namely SMOSA (Serafini, 1992), UMOSA (Ulungu et al., 1999), CMOSA (Czyzak and Jaszkiewicz, 1998) and three MOMA-like algorithms, namely MOEA/D (Zhang and Li, 2007), MOGLS (Jaszkiewicz, 2002), and IMMOGLS (Ishibuchi et al., 2003). All algorithms were implemented in C++. We ran all experiments on a PC Pentium CPU (Duo Core) 1.73 GHZ with 2GB memory.

6.1 Performance Indicators

The performance assessment of various algorithms is one of the most important issues in the EMO research. It is well established that the quality (in the objective space) of approximation sets should be evaluated on the basis of two criteria: the closeness to the POF (convergence) and the wide and even coverage of the POF (diversity). In this paper, we use the following two popular indicators for comparing the performance of the multi-objective metaheuristics under consideration.

6.1.1 Inverted Generational Distance (IGD-metric)

This indicator (see Coello Coello and Cruz Cortés (2005)) measures the average distance from a set of reference points P^* to the approximation set P. It can be formulated as

Table 1: Test instances of MOKP and MOTSP and computational cost.

MOKP				MOTSP			
Instance	n	m	# nfes	Instance	n	m	# nfes
KS2502	250	2	75000	KROAB50	50	2	500000
KS5002	500	2	100000	KROCD50	50	2	500000
KS7502	750	2	125000	KROAB100	100	2	2500000
KS2503	250	3	100000	KROCD100	100	2	2500000
KS5003	500	3	125000	KROABC100	100	3	5000000
KS7503	750	3	150000	KROBCD100	100	3	5000000

Table 2: The setting of population size in MOSA-like algorithms.

MOKP				MOTSP			
Instance	EMOSA	UMOSA	CMOSA	Instance	EMOSA	UMOSA	CMOSA
KS2502	50	100	100	KROAB50	50	100	100
KS5002	50	100	100	KROCD50	50	100	100
KS7502	50	100	100	KROAB100	100	100	100
KS2503	50	300	300	KROCD100	100	100	100
KS5003	50	300	300	KROABC100	100	100	100
KS7503	50	300	300	KROBCD100	100	100	100

IGD$(P, P^*) = \frac{1}{|P^*|} \sum_{u \in P^*} \min_{v \in P} \text{dist}(u, v)$. Ideally, the reference set should be the whole true POF. Unfortunately, the true POF of many MOCO problems is unknown in advance. Instead, a very good approximation to the true POF can be used. In this paper, the reference sets P^* are formed by collecting all non-dominated solutions obtained by all algorithms in all runs. The smaller the IGD-metric value, the better the quality of the approximation set P. To have a lower value of the IGD-metric, the obtained solutions should be close to P^* and should not miss any part of P^*. Therefore, the IGD-metric can assess the quality of approximation sets with respect to both convergence and diversity.

6.1.2 Hypervolume (S-metric or Lebesgue measure)

This indicator (see Zitzler and Thiele (1999)) measures the volume enclosed by the approximation set P wrt a reference point $r = (r_1, \ldots, r_m)$. In the case of minimization, this metric can be written as $S(P, r) = \text{VOL}\left(\bigcup_{u \in P} E(u, r)\right)$, where $E(u, r) = \{v | u_i \leq v_i \leq r_i, i = 1, \ldots, m\}$ (i.e. $E(u, r)$ is the hyperplane between two m-dimensional points u and r)[2]. Compared to the reference set required in the IGD-metric, establishing the reference point in the S-metric is relatively easier and more flexible. In our experiments, r can be set to be a point dominated by the nadir point: $r_i = z_i^{nad} + 0.5(z_i^{nad} - z_i^{ideal}), i = 1, \ldots, m$. The larger the S-metric value, the better the quality of the approximation set P, which means a smaller distance to the true POF and a better distribution of the approximation set. Similar to the IGD-metric, better S-metric values can be achieved when the obtained solutions are close to the POF and achieve a full coverage of the POF. Therefore, the S-metric also estimates the quality of approximation sets both in terms of convergence and diversity.

6.2 EMOSA vs. MOSA-like Algorithms

6.2.1 Experimental Settings

In this first set of experiments, we used the test instances with two and three objectives shown in Table 1. We used the same following parameters in the annealing schedule of the four algorithms: $T_{max} = 1.0, T_{min} = 0.01, T_{reheat} = 0.1, \alpha_1 = 0.8$, and $\alpha_2 = 0.5$. Moreover, all four algorithms were allocated the same number of function evaluations

[2]In the case of maximization, $E(u, r) = \{v | r_i \leq v_i \leq u_i, i = 1, \ldots, m\}$.

Table 3: Mean and standard deviation of IGD-metric values on MOKP instances.

Instance	EMOSA	UMOSA	CMOSA	SMOSA
KS2502	**16.2** (1.86)	34.0 (2.23)	223.5 (19.54)	30.6 (2.17)
KS5002	**23.1** (1.33)	64.9 (2.31)	382.6 (42.57)	80.5 (3.42)
KS7502	**32.2** (2.10)	102.2 (3.75)	657.1 (68.96)	145.2 (6.54)
KS2503	**54.2** (1.10)	71.0 (1.08)	115.2 (5.17)	78.9 (1.45)
KS5003	**86.4** (1.40)	129.5 (1.80)	325.0 (16.37)	184.6 (5.12)
KS7503	**110.7** (2.56)	193.8 (3.76)	547.7 (36.66)	274.0 (4.68)

Table 4: Mean and standard deviation of of S-metric values on MOKP instances.

Instance	EMOSA	UMOSA	CMOSA	SMOSA
KS2502	**1.27E+07** (1.17E+04)	1.26E+07 (0.64E+04)	1.16E+07 (8.33E+04)	1.26E+07 (0.93E+04)
KS5002	**3.55E+07** (0.17E+05)	3.51E+07 (0.27E+05)	3.16E+07 (2.97E+05)	3.50E+07 (0.36E+05)
KS7502	**1.04E+08** (0.46E+05)	1.03E+08 (0.56E+05)	9.30E+07 (10.5E+05)	1.03E+08 (0.87E+05)
KS2503	**5.83E+10** (0.49E+08)	5.77E+10 (0.51E+08)	5.55E+10 (1.93E+08)	5.73E+10 (0.62E+08)
KS5003	**4.16E+11** (0.29E+09)	4.11E+11 (0.26E+09)	3.80E+11 (1.69E+09)	4.03E+11 (5.52E+09)
KS7503	**1.26E+12** (0.85E+11)	1.25E+12 (0.63E+11)	1.11E+12 (13.4E+11)	1.21E+12 (1.93E+11)

in each of 20 runs for the same test instance as detailed in column # nfes of Table 1. The population sizes for the three population-based MOSA-like algorithms are given in Table 2. We use a smaller (than in UMOSA and CMOSA) population size in EMOSA for most of the test instances. The reason for this is that EMOSA has the ability to search different parts of the POF by adapting weight vectors. Note that SMOSA is a single point based search method.

At every temperature level, the number #ls of neighbors examined by local search is set to 10 for each MOKP instance and 250 for each MOTSP instance. It should be noted that the total number of neighbors examined in SMOSA at each temperature level equals the total number of neighbors examined for the whole population in UMOSA and CMOSA at the same temperature level. For all test instances in both problems, the neighborhood size in EMOSA is set to $K = 10$. In both SMOSA and CMOSA, each component of the weight vectors is increased or decreased by $2.0/Q$ randomly or using the heuristic in eq. (4).

6.2.2 Experimental Results

The mean and standard deviation values of the IGD-metric and S-metric found by the four algorithms are shown in Tables 3 and 4, respectively. From these results, it is evident that EMOSA outperforms the three other MOSA-like algorithms in terms of both performance indicators on all six MOKP test instances. The superiority of EMOSA is due to the stronger selection pressure to move towards the POF by having competition among individuals. We believe that this competition allows the local search in EMOSA to start from a high-quality initial solution at every temperature level.

We can also observe from Tables 3 and 4 that CMOSA is clearly inferior to the three other MOSA-like algorithms. This is due to the fact that CMOSA modifies the weight vector of each individual while examining every neighboring solution. This change of weight vectors in the early stage of the multi-objective search does not affect the distribution of the non-dominated solutions because the current population is still distant to the POF. However, the frequent change of weight vectors could potentially guide each individual to move along different search directions in the objective space. Of course, this will slow down the convergence speed towards the POF. This is why our proposed EMOSA approach only adapts the weight vector of each individual once the temperature is below T_{min}.

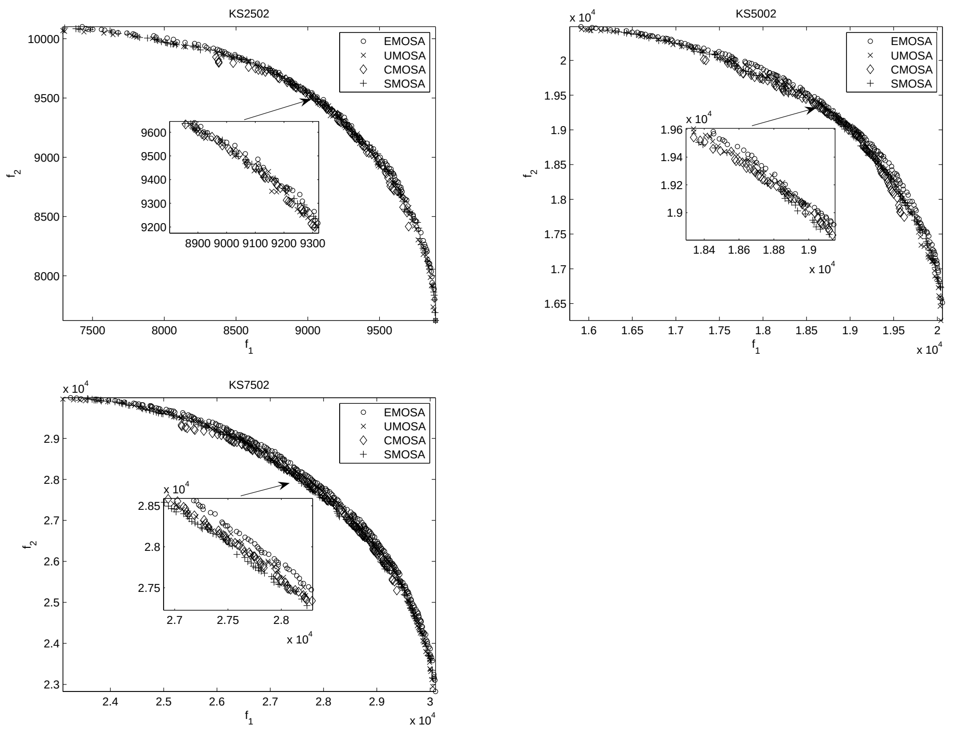


Figure 6: Non-dominated solutions found by the four MOSA-like algorithms on three bi-objective knapsack instances.

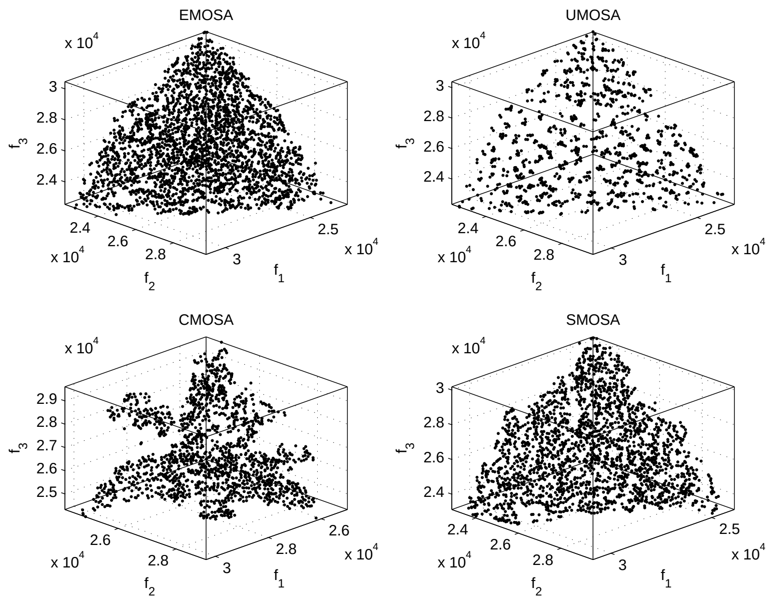


Figure 7: Non-dominated solutions found by the four MOSA-like algorithms on the three-objective knapsack instance KS7503.

Table 5: Mean and standard deviation of IGD-metric values on MOTSP instances.

Instance	EMOSA	UMOSA	CMOSA	SMOSA
KROAB50	**781.20** (184.6)	1427.1 (233.7)	12409 (618.3)	6385.5 (447.8)
KROCD50	**751.54** (132.0)	1405.0 (129.9)	11696 (672.0)	6279.6 (392.4)
KROAB100	**1607.1** (234.4)	4327.0 (311.2)	36191 (733.5)	24426 (676.7)
KROCD100	**1464.8** (217.2)	4032.7 (171.3)	35588 (931.8)	23227 (872.3)
KROABC100	**4421.5** (173.3)	5126.9 (143.6)	29819 (659.4)	38646 (473.6)
KROBCD100	**4253.3** (134.8)	4972.0 (150.2)	29008 (626.2)	37269 (460.1)

Table 6: Mean and standard deviation of S-metric values on MOTSP instances.

Instance	EMOSA	UMOSA	CMOSA	SMOSA
KROAB50	**7.46E+11** (5.32E+10)	5.44E+11 (3.83E+10)	2.43E+11 (2.72E+10)	3.00E+11 (3.31E+10)
KROCD50	**9.83E+11** (5.96E+10)	7.88E+11 (3.65E+10)	3.47E+11 (3.0E+10)	4.33E+11 (4.16E+10)
KROAB100	**4.25E+12** (2.66E+11)	3.71E+12 (2.58E+11)	2.10E+12 (1.42E+11)	2.06E+12 (2.42E+11)
KROCD100	**4.00E+12** (2.39E+11)	3.45E+12 (3.09E+11)	1.84E+12 (1.05E+11)	1.77E+12 (1.68E+11)
KROABC100	**9.12E+20** (7.0E+19)	7.32E+20 (4.57E+19)	3.06E+20 (2.86E+19)	1.40E+20 (1.79E+19)
KROBCD100	**9.12E+20** (6.02E+10)	7.35E+20 (4.65E+10)	2.95E+20 (2.81E+19)	1.34E+20 (2.14E+19)

Figure 6 shows the non-dominated solutions found by the four algorithms on three bi-objective MOKP instances for the run for which each algorithm achieved its best value for the IGD-metric. We can see that EMOSA clearly found better solutions than the other three MOSA-like algorithms on KS5002 and KS7502. For the smallest instance KS2502, EMOSA is only slightly better. Moreover, both UMOSA and SMOSA perform quite similarly on these three instances while CMOSA has the tendency to locate the non-dominated solutions close to the middle part of the POF.

In Figure 7, we plot the non-dominated solutions obtained by the four algorithms on the largest MOKP instance with three objectives (KS7503) in the run for which each algorithm achieved its best value for the IGD-metric. It is clear that EMOSA produced the non-dominated front with the best diversity while the non-dominated front by CMOSA has the worst distribution. This is because EMOSA tunes the weight vectors according to distances of both objective vectors and weight vectors to maintain the diversity of the set of current weight vectors, which are less likely to be very similar. In contrast, CMOSA is unable to preserve the diversity of weight vectors particularly in the case of test instances with more than two objectives.

From Figure 7, we can also note that the set of non-dominated solutions found by UMOSA consists of many isolated clusters. This is because this MOSA-like algorithm with fixed weights has no ability to approximate all parts of the POF when the size of this front is very large. However, it can still produce a good approximation to the POF. This is why EMOSA uses fixed weights only in the early stage of its search process. It should be pointed out that the population size used in UMOSA and CMOSA is six times that of EMOSA for all 3-objective instances (50 vs. 300). This also demonstrates the efficiency of the strategy for adapting weights in EMOSA which requires a considerably smaller population size.

Tables 5 and 6 show the mean and standard deviation values of the IGD-metric and S-metric obtained by the four MOSA-like algorithms on the six MOTSP instances. Again, EMOSA performs better than the other three MOSA-like algorithms on all six MOTSP test instances in terms of the IGD-metric and the S-metric. In contrast, CMOSA shows again the worst performance.

Figure 8 shows the distributions of the non-dominated solutions obtained by the four MOSA-like algorithms in the run for which each algorithm achieved its best value for the IGD-metric value. It is clear that EMOSA found better non-dominated solu-

Table 7: The average running time (seconds) of four MOSA-like algorithms on two 100-city MOTSP instances.

Instance	EMOSA	UMOSA	CMOSA	SMOSA
KROAB100	6.7	3.4	31.2	13.7
KROABC100	169.3	127.4	142.0	39.0

tions in terms of both convergence and diversity. Figure 9 shows the non-dominated solutions obtained in the best run of each algorithm for the three-objective instance - KROABC100. It can be seen that the diversity of the solutions found by EMOSA is the best among the four algorithms.

From all above results, we can also conclude that the overall performance of UMOSA is better than that of CMOSA and SMOSA for both problems. These results are also consistent with those reported in previous studies (Banos et al., 2007). The poor performance of CMOSA and SMOSA is caused by the frequent modification of weight vectors, which might slow down the convergence speed.

The four MOSA-like algorithms use different strategies to maintain the diversity of weight vectors. The running times of these algorithms rely on the computational complexity of the related strategies. Table 7 shows the average computational time used by the four MOSA-like algorithms on two 100-city MOTSP instances. From this table, we can make the following observations:

- For the smaller 2-objective instance KROAB100, UMOSA is the fastest algorithm while CMOSA is the slowest. This is because UMOSA maintains a set of fixed weight vectors which does not have a computational cost because it is not required to tune the weight vectors. In contrast, both CMOSA and SMOSA modify weight vectors at each single local search move. The computational complexity of the strategy in CMOSA is much higher than that in SMOSA, however EMOSA is still faster. Therefore, it seems that it is not reasonable to tune the weight vectors at every local search move.
- For the larger 3-objective instance KROABC100, both EMOSA and UMOSA are slower than SMOSA. This is contrary to the above observation. The reason for this is that the the number of non-dominated solutions found by the former two algorithms is much larger than those found by SMOSA. In this case, most of the computational cost is allocated to maintaining the external population. This is also the reason why CMOSA is faster than EMOSA.

6.3 EMOSA vs. MOMA-like Algorithms

6.3.1 Experimental Settings

In this section, we compare EMOSA to three MOMA-like algorithms, namely MOEA/D, MOGLS, and IMMOGLS. All parameter settings used in EMOSA are the same as in the previous section. The population sizes used in the other three algorithms are given in Table 8. Like EMOSA, the neighborhood size used in MOEA/D is also 10 for all test instances. In MOGLS, the size of the mating pool is set to 20 for every test instance. In IMMOGLS, 10 elite solutions are randomly selected from the external

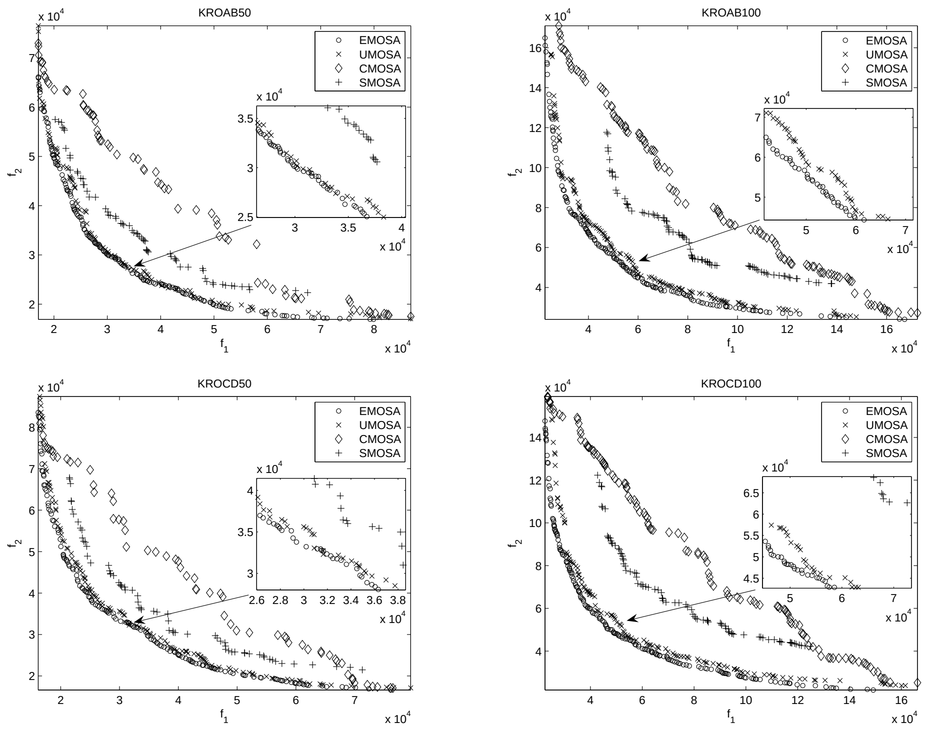


Figure 8: The non-dominated solutions found by MOSA-like algorithms on bi-objective MOTSP instances.

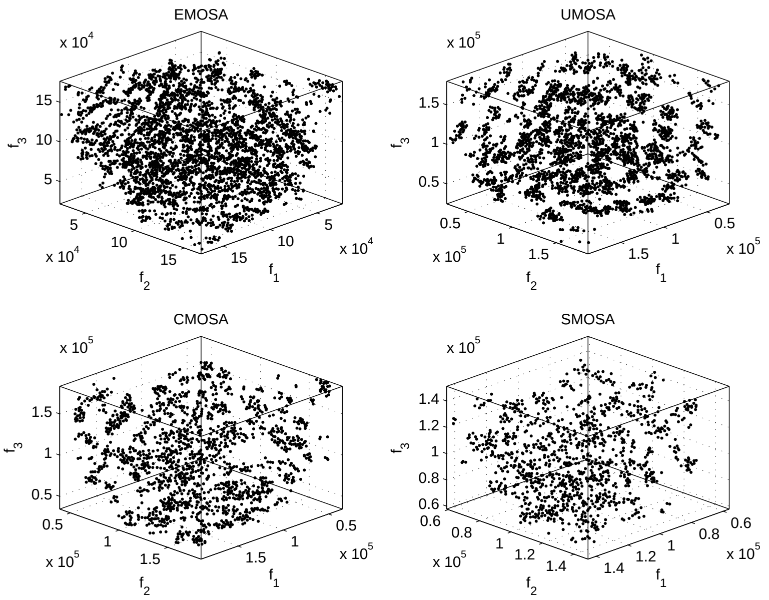


Figure 9: The non-dominated solutions found by MOSA-like algorithms on instance KROABC100.

Table 8: The population size used in the MOMA-like algorithms.

MOKP				MOTSP			
Instance	MOEA/D	MOGLS	IMMOGLS	Instance	MOEA/D	MOGLS	IMMOGLS
KS2502	100	100	100	KROAB50	100	100	100
KS5002	100	100	100	KROCD50	100	100	100
KS7502	100	100	100	KROAB100	100	100	100
KS2503	300	300	300	KROCD100	100	100	100
KS5003	300	300	300	KROABC100	100	100	100
KS7503	300	300	300	KROBCD100	100	100	100

Table 9: Mean and standard deviation of IGD-metric values found by MOMA-like algorithms on MOKP instances.

Instance	EMOSA	MOEA/D	MOGLS	IMMOGLS
KS2502	**16.2** (1.86)	24.0 (4.42)	21.5 (4.70)	100.5 (15.23)
KS5002	**23.1** (1.33)	64.3 (5.82)	55.6 (2.88)	292.8 (29.68)
KS7502	**32.2** (2.10)	157.5(10.87)	134.6(9.08)	620.3 (40.70)
KS2503	**54.2** (1.10)	66.6 (1.68)	56.4 (2.0)	252.1 (23.12)
KS5003	**86.4** (1.40)	166.6(4.11)	134.8(3.84)	734.1 (59.79)
KS7503	**110.7**(2.56)	334.5(12.77)	233.3(5.20)	1160.7(44.70)

population. For the MOKP problem, single-point crossover is performed, and each bit of the binary string is mutated by flipping with probability 0.01. The neighborhood structures for both the MOKP problem and the MOTSP problem are the same as in the previous section. For the MOTSP instances, offspring solutions are sampled by using cycle crossover (Larranaga et al., 1999) in these three MOMA-like algorithms.

6.3.2 Experimental Results

The mean and standard deviation of the IGD-metric and S-metric values found by EMOSA and the three MOMA-like algorithms on six MOKP test instances are shown in Table 9 and Table 10, respectively. From this set of results, it is evident that EMOSA performs better than the three MOMA-like algorithms on all test instances. We think that the superiority of EMOSA is due to i) the use of a diverse set of weight vectors and ii) the use of simulated annealing for local search.

Among the four algorithms, IMMOGLS has the worst performance on all test instances. This is because IMMOGLS uses a fixed population size and updates the current population with all offspring solutions without competition. In EMOSA, individuals with similar weight vectors interact with each other via competition, which helps for a good performance of EMOSA in terms of convergence.

Zhang and Li (2007) showed that MOEA/D constantly performs better than MOGLS on the MOKP test instances when local search heuristic is applied. When simple hill-climber local search is used in both algorithms, MOEA/D performs slightly worse than MOGLS regarding the IGD-metric since the population size is not large

Table 10: Mean and standard deviation of S-metric values found by MOMA-like algorithms on MOKP instances.

Instance	EMOSA	MOEA/D	MOGLS	IMMOGLS
KS2502	**1.27E+07**(1.17E+04)	1.27E+07(1.34E+04)	1.27E+07(1.12E+04)	1.21E+07(7.73E+04)
KS5002	**3.55E+07**(0.18E+05)	3.51E+07(0.58E+05)	3.51E+07(0.44E+05)	3.21E+07(3.02E+05)
KS7502	**1.04E+08**(0.46E+05)	1.02E+08(1.42E+05)	1.02E+08(1.56E+05)	9.14E+07(10.6E+05)
KS2503	**5.83E+10**(0.49E+08)	5.80E+10(0.59E+08)	5.77E+10(1.10E+08)	4.90E+10(9.31E+08)
KS5003	**4.16E+11**(0.29E+09)	4.07E+11(0.55E+09)	4.04E+11(1.09E+09)	3.10E+11(5.51E+09)
KS7503	**1.26E+12**(0.85E+09)	1.20E+12(2.37E+09)	1.20E+12(3.57E+09)	8.87E+11(15.1E+09)

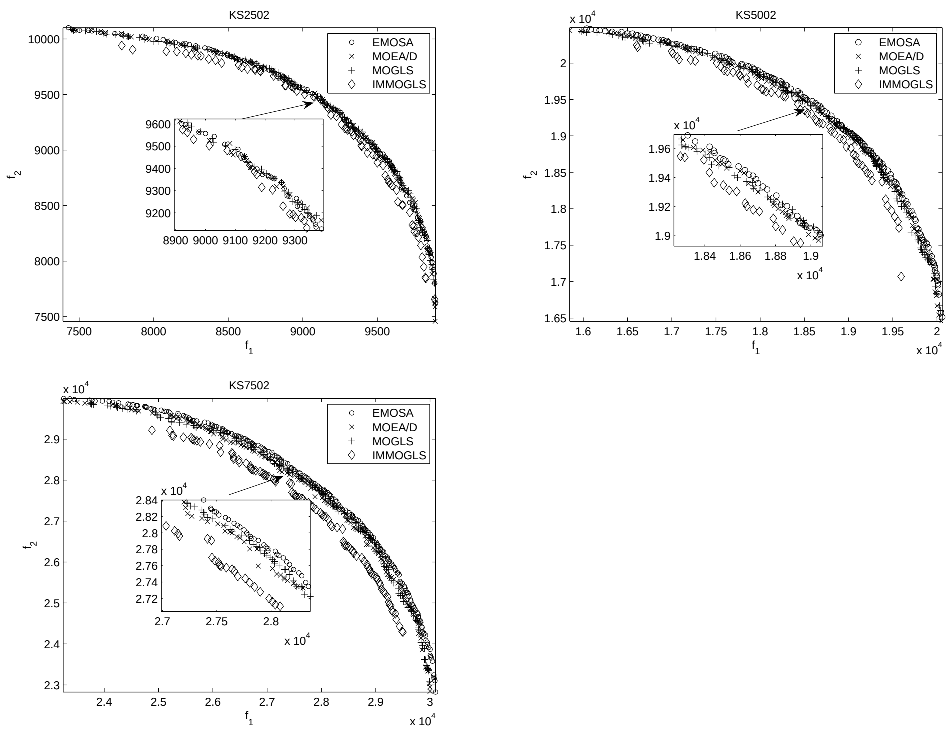


Figure 10: Non-dominated solutions found by MOMA-like algorithms on bi-objective knapsack instances.

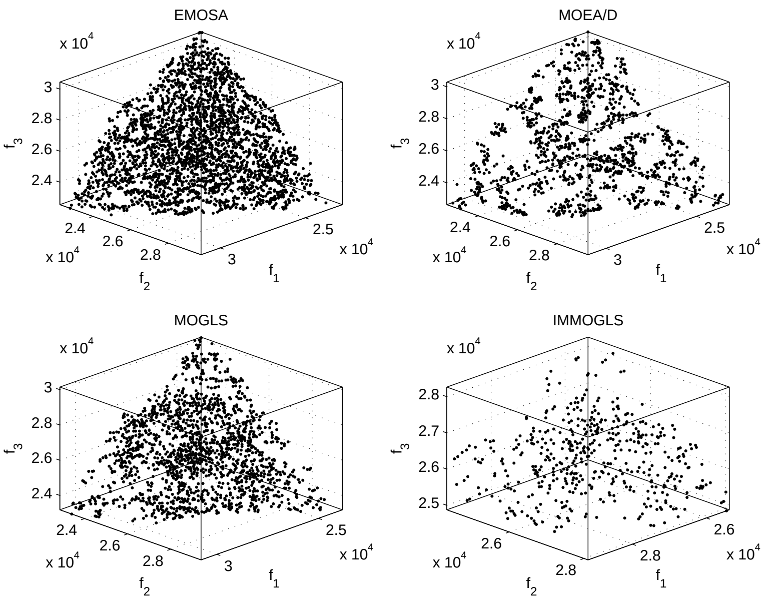


Figure 11: Non-dominated solutions found by MOMA-like algorithms on the KS7503 instance.

Table 11: Mean and standard deviation of IGD-metric values found by MOMA-like algorithms on MOTSP instances.

Instance	EMOSA	MOEA/D	MOGLS	IMMOGLS
KROAB50	**781.2**(184.6)	1253.8(219.7)	3574.7(620.7)	8412.4(923.9)
KROCD50	**751.5**(132.0)	1220.9(201.2)	3744.7(430.4)	8182.6(851.6)
KROAB100	**1607.1**(234.4)	2184.5(258.7)	17100(1524)	24052(1156)
KROCD100	**1464.8**(217.2)	2030.9(307.2)	16337(930)	22840(2050)
KROABC100	**4421.5**(173.3)	6126.6(254.8)	22789(868)	34419(831.6)
KROBCD100	**4253.3**(134.8)	6189.3(236.4)	22079(963)	33475(1291.4)

Table 12: Mean and standard deviation of S-metric values found by four MOMA algorithms on six MOTSP instances.

Instance	EMOSA	MOEA/D	MOGLS	IMMOGLS
KROAB50	**7.46E+11**(5.33E+10)	5.98E+11(5.44E+10)	3.62E+11(7.07E+10)	1.97E+11(2.72E+10)
KROCD50	**9.83E+11**(5.96E+10)	8.31E+11(5.68E+10)	5.02E+11(6.69E+10)	3.02E+11(5.12E+10)
KROAB100	**4.25E+12**(2.66E+11)	3.37E+12(3.11E+10)	1.84E+12(1.62E+11)	1.59E+12(2.25E+11)
KROCD100	**4.01E+12**(2.39E+11)	3.20E+12(2.90E+11)	1.49E+12(2.36E+11)	1.35E+12(2.06E+11)
KROABC100	**9.12E+20**(7.00E+19)	2.63E+20(1.80E+19)	1.32E+20(2.75E+19)	8.82E+19(2.17E+19)
KROBCD100	**9.12E+20**(6.03E+19)	2.45E+20(1.99E+19)	1.13E+20(2.70E+19)	7.91E+19(2.03E+19)

enough, but better in terms of the S-metric. In contrast, EMOSA clearly outperforms MOGLS, which gives an indication that the use of advanced local search based metaheuristics can enhance the performance of EMO algorithms.

The final solutions found by EMOSA and the three MOMA-like algorithms on three 2-objective MOKP instances are plotted in Figure 10. For KS2502, EMOSA performs similarly to MOEA/D and MOGLS. However, the solutions found by EMOSA are better than those obtained by MOEA/D and MOGLS on two larger test instances KS5002 and KS7502. It is clear that the solutions of IMMOGLS are dominated by those of the other three algorithms.

Figure 11 shows the solutions found by the four algorithms on instance KS7503 for the run in which each algorithm achieved the best IGD-metric value. It is evident that EMOSA has the best performance wrt diversity. In MOEA/D, the non-dominated solutions are located in a number of clusters. This is because MOEA/D uses fixed weights during the whole search process. Therefore, the performance of MOEA/D depends on the population size. For large test instances, a reasonable large number of weight vectors is required to achieve good diversity in the final non-dominated set of solutions. However, although the population size in EMOSA is small, this algorithm is still capable of solving large test instances due to the effective adaptation of weight vectors.

Table 11 and Table 12 give the mean and standard deviation of IGD-metric and S-metric values found by EMOSA and the three MOMA-like algorithms on six MOTSP test instances. From these experimental results, it is clear that EMOSA found the best solutions in terms of both indicators. Figure 12 shows the distribution of the solutions obtained by the four algorithms on four bi-objective MOTSP test instances. As we can see, both EMOSA and MOEA/D have good performance in terms of convergence. The solutions found by both algorithms are quite close to each other. As for diversity, the distribution of the solutions found by EMOSA is more uniform. Both MOGLS and IMMOGLS have worse performance than EMOSA in all test instances. This could be due to the lack of efficient genetic operators for sampling promising offspring solutions for multi-objective local search.

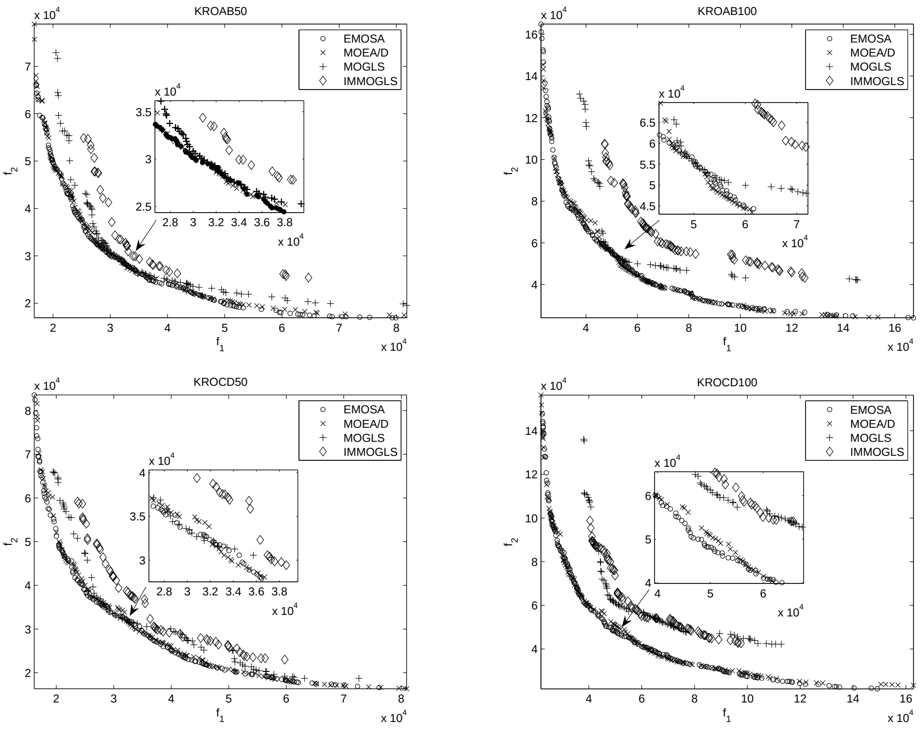


Figure 12: Non-dominated solutions found by MOMA-like algorithms on bi-objective TSP instances.

Figure 13 shows the distribution of the non-dominated solutions found by the four algorithms on the 3-objective KROABC100 instance. Unlike the concave POF of instance KS7503, the POF of instance KROABC100 is convex. We can observe that EMOSA has the best performance in finding a good distribution of non-dominated solutions.

7 Effect of Algorithmic Components and Analysis of Parameter Sensitivity in EMOSA

7.1 Effect of Algorithmic Components

7.1.1 Adaptation of Weight Vectors

In our early work on EMOSA (Li and Landa-Silva, 2008), we tried the same strategy as in CMOSA to adapt weight vectors. As we discussed in section 3, that strategy has some weaknesses. In this section, we demonstrate the improvement of EMOSA due to the new diversity strategy. We modified the EMOSA approach shown in **Procedure** 1 by replacing our weights adaptation strategy with the strategy used in CMOSA and applied the modified algorithm to instance KROABC100. All parameters in this modified EMOSA remain the same as those in section 6.2.1. Figure 14 shows the non-dominated solutions found by this modified EMOSA. Compared to the results in Figure 13, the modified EMOSA performs worse in terms of diversity. This indicates that our new di-

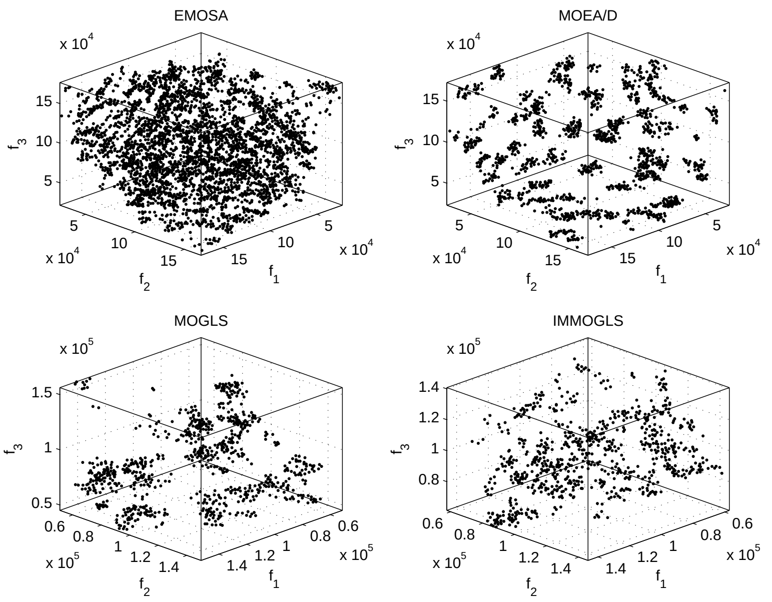


Figure 13: Non-dominated solutions found by MOMA-like algorithms on the KROABC100 instance.

versity strategy works better in guiding the multi-objective search towards unexplored parts of the POF by maintaining the diversity of weight vectors.

7.1.2 Competition Strategies

As we said before, EMOSA distinguishes itself from other algorithms by the adaptive competition scheme between individuals. We carried out three further experiments to study the effect of the competition among individuals on the performance of EMOSA.

To study the effect of weighted aggregation function on the convergence of EMOSA, we compared two versions of EMOSA, one using the weighted sum approach and the other one using the weighted Tchebycheff approach. Both versions use the

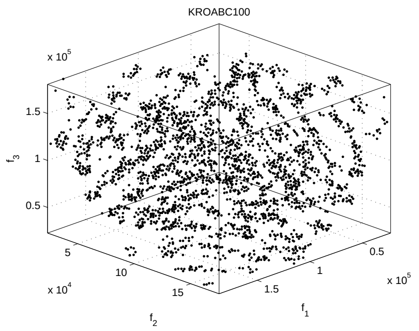


Figure 14: CMOSA's diversity strategy.

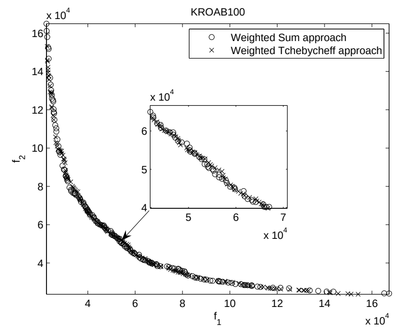


Figure 15: Weighted approaches.

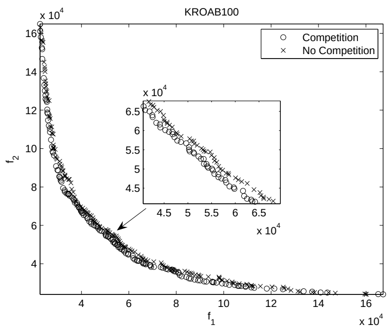


Figure 16: Effect of competition.

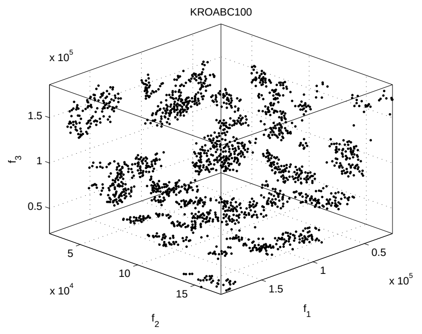


Figure 17: No Pareto selection.

same parameter settings described in Section 6.2.1. Figure 15 shows the non-dominated solutions found by these two versions. From these results, we can observe that the weighted Tchebycheff approach performs very similar to the weighted sum approach on KROAB100. Zhang and Li (2007) made a different observation, that the former has worse performance on the MOKP problem. The reason for this is that the weighted sum approach has higher effectiveness in preserving the quality of infeasible solutions in the repair heuristic.

Also, we tried the version of EMOSA with no competition between individuals. That is, the current solution of each weighted aggregation function does not compete with its neighboring solutions. To implement this, lines 4-8 in **Procedure 5** were removed. We tested this version of EMOSA on the KROAB100 instance using the same parameters as in section 6.2.1. Figure 16 shows the non-dominated solutions found by EMOSA without competition between individuals. Clearly, the convergence of this version is worse than the EMOSA with competition.

Moreover, we tested another version of EMOSA with competition based on only scalarization. In **Procedure 5**, lines 5-7 were removed. Instead, each neighboring solution $x^{(k)}$ is replaced with y if $g(y, \lambda^{(k)}) < g(x^{(k)}, \lambda^{(k)})$. This variant of EMOSA was tested on instance KROABC100. The final solutions found by this variant are plotted in Figure 17. It can be observed that the modified EMOSA failed to find a set of well-distributed non-dominated solutions despite having the adaptation of weight vectors. As mentioned before, this is caused by the loss of diversity in the population. When the current population is close to the POF, neighboring solutions are very likely to be mutually non-dominated. However, one solution can still be replaced by its neighbors in terms of a weighted sum function. In this case, EMOSA might miss the chance to approximate the part of POF nearby such a solution. This is why EMOSA does not work well on KROABC100 wrt diversity.

7.1.3 Choice of Neighborhood Structure

The choice of appropriate neighborhood structure plays a very important role in local search methods for combinatorial optimization. In this paper, we have proposed a k-bit insertion neighborhood structure for the MOKP problem. In this section, we compare two versions of EMOSA on the instance KS2502, one using the k-bit insertion neighborhood and another version using the k-bit flipping neighborhood. Figure 18 shows the

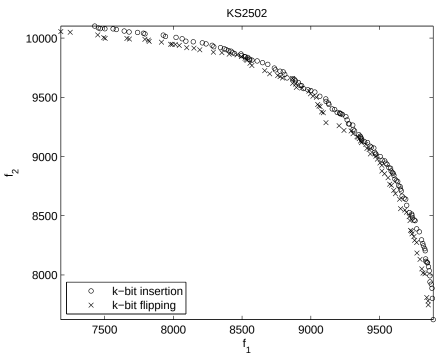


Figure 18: k-bit insertion vs. k-bit flipping.

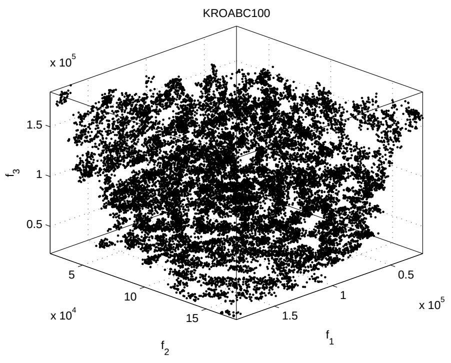


Figure 19: External population update.

final solutions found by both versions in the best of 20 runs regarding the IGD-metric. It can be observed that the non-dominated solutions obtained by EMOSA with the k-bit insertion neighborhood dominate all solutions produced by the version with the k-bit flipping neighborhood. Moreover, the mean IGD-metric value obtained when using the k-bit flipping neighborhood is 76.934, which is worse than the value obtained when using the k-bit insertion (16.245).

7.1.4 Updating the External Population with ϵ-dominance

In MOEA/D, no diversity strategy is applied to manage the external population. This is not a problem when the size of the POF is small. However, for large fronts, the external population might contain more and more non-dominated solutions as the search progresses. The computational time needed to update the external population may exceed substantially the time used by all other components of EMOSA. This is why we incorporate ϵ-dominance into EMOSA to help in keeping the diversity of EP automatically. Figure 19 shows the results obtained by EMOSA without using ϵ-dominance on instance KROABC100. Compared to the results in Figure 9, the distribution of non-dominated solutions is more dense that the distribution obtained with EMOSA using ϵ-dominance. Up to 20,000 non-dominated solutions are saved in the external population while EMOSA with ϵ-dominance reports only about 3000 solutions at the end of each run. In terms of the computational time, we also noticed that EMOSA without ϵ-dominance used about 1600 seconds to complete one run, while EMOSA with ϵ-dominance only spent less than 170 seconds to complete a single run, which represents almost a ten-fold reduction.

7.2 Analysis of Parameter Sensitivity

7.2.1 Parameters in Two-stage Cooling Scheme

Like the simulated annealing algorithms for single objective optimization, the performance of EMOSA might also be influenced by the setting of parameters in the cooling scheme. To verify this, we studied two parameters T_{reheat} (reheat temperature value) and α_2 (fast cooling rate). EMOSA with the combinations of $T_{reheat} = \{0.1, 0.3, 0.5, 0.8, 1.0\}$ and $\alpha_2 = \{0.3, 0.5, 0.8\}$ were tested on KROAB50 and KROAB100 in 20 runs. Figure 20 plots the mean IGD-metric values found by EMOSA with differ-

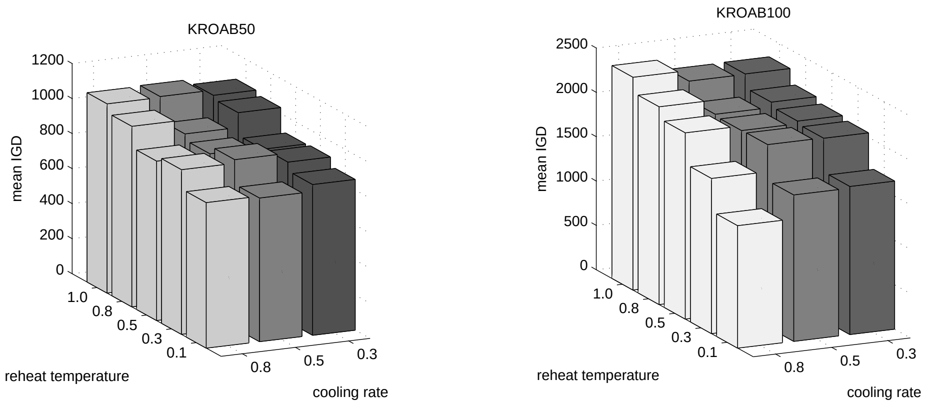


Figure 20: The mean IGD-metric values found by EMOSA with 15 combinations of T_{reheat} and α_2.

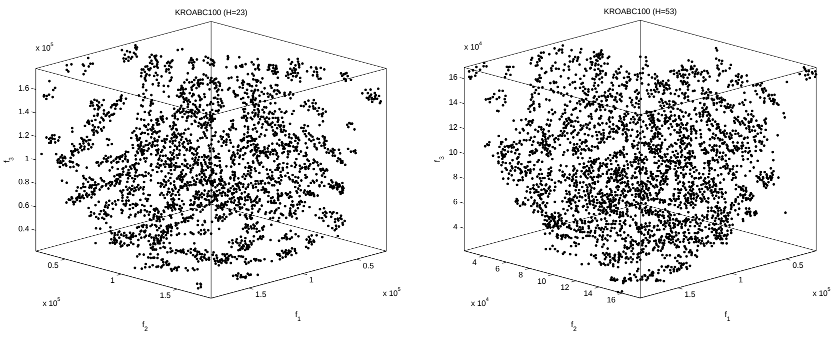


Figure 21: The non-dominated solutions found by EMOSA with $H = 23$ (left) and $H = 53$ (right) for KROABC100.

ent combinations of T_{reheat} and α_2. It can be observed that EMOSA with lower reheat temperature has better performance. For the smaller instance KROAB50, the best result was found by EMOSA using a faster cooling rate (0.5). However, such a strategy does not perform best for KROAB100. For this larger instance, EMOSA with a slower cooling rate (0.8) worked better. For the small instance, the current population of EMOSA is very likely to be well-converged towards the Pareto front. In this case, EMOSA needs to intensify the search near the Pareto front. Therefore, the temperature levels in EMOSA should be in lower values. In contrast, EMOSA with slow cooling rate is able to increase the diversity of the search for large instances.

7.2.2 Population Size (Q) and the Number of Uniform Weight Vectors (H)

In EMOSA, the set of current weight vectors are chosen from a set of predefined uniform weight vectors. Such a set is controlled by a parameter H. In this section, we study the influence of H on the performance of EMOSA. We tested EMOSA with a smaller H=23 (300 uniform weight vectors) and a larger H=53 (1485 uniform weight vectors) for KROABC100. Figure 21 shows the distribution of non-dominated solutions found by EMOSA with $H = 23$ and $H = 53$. We can observe from this figure that

EMOSA with the larger value of H performed better than that with the smaller value of H wrt diversity. It is easy to understand that more uniform weight vectors are needed in EMOSA when the size of Pareto front is very large. However, a very large value of H will increase the computational time of adapting weight vectors in EMOSA.

As we mentioned before, EMOSA produced better results than other MOMA-like algorithms although its population size is much smaller. The main reason for this is that EMOSA uses a strategy to adapt weight vectors. We also tested EMOSA with population sizes of 100, 200, 300, 400 and 500 on instance KROABC100. Correspondingly, the obtained mean IGD values were 4308.13, 4078.63, 4396.37, 5280.28 and 5792.63. It is clear that EMOSA with population sizes smaller than 300 found similar IGD-metric values. In contrast, EMOSA with larger population sizes performed worse.

8 Conclusions

Both evolutionary algorithms and simulated annealing are popular stochastic search techniques for tackling multi-objective combinatorial optimization. However, the hybridization of both methods has not yet been studied in the context of MOCO problems. Most of the existing research work on multi-objective memetic algorithms only focuses on the use of simple hill-climber local search within evolutionary algorithms. In this paper, we proposed an adaptive evolutionary multi-objective approach combined with simulated annealing. We call this approach Evolutionary Multi-objective Simulated Annealing (EMOSA). The algorithm seeks to optimize multiple weighted aggregation functions. To approximate various parts of the POF, we have suggested a new method to tune the weight vectors of these aggregation functions. Moreover, ϵ-dominance was used to update the external population of non-dominated solutions in EMOSA for enhancing computational efficiency.

We also compared EMOSA to six other algorithms using weighted aggregation (three MOMA-like algorithms and three MOSA-like algorithms) on both the multi-objective knapsack problems and the multi-objective traveling salesman problem. Our experimental results show that EMOSA performs better than all other algorithms considered in terms of both the IGD-metric (inverted generational distance) and the S-metric (hypervolume). In particular, EMOSA is capable of finding a very good distribution of non-dominated solutions for the 3-objective test instances of both problems. The main reason for this is that our new strategy for tuning weight vectors directs the search towards different portions of the POF in a more effective way than the old strategy. Moreover, we also studied here the effects of important components on the performance of the proposed EMOSA algorithm. We have shown that the use of ϵ-dominance significantly reduces the computational time for maintaining the external population of non-dominated solutions.

It should be noted that the POF shapes of all test instances in both MOCO problems considered in this paper do not present difficulties to EMOSA using the weighted sum approach. As future work, in order to deal with some hard POF shapes, such as discontinuity, we intend to design a robust version of EMOSA to tackle such MOCO problems. Moreover, we also intend to investigate the ability of EMOSA to solve other challenging MOPs, such as continuous MOPs with many local optima, MOCO problems with complex constraints, many-objective problems, and dynamic MOPs. The C++ source code of EMOSA is available from the authors at request.

References

Banos, R., Gil, C., Paechter, B., and Ortega, J. (2007). A hybrid meta-heuristic for multi-objective optimization: MOSATS. *Journal of Mathematical Modelling and Algorithms*, 6(2):213–230.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308.

Burke, E., Cowling, P., and Landa Silva, J. (2001). Hybrid population-based metaheuristic approaches for the space allocation problem. In *Proc. of the 2001 Congress on Evolutionary Computation (CEC 2001)*, pages 232–239, Seoul, Korea. IEEE press.

Burke, E. K. and Kendall, G., editors (2005). *Search methodologies - Introductory Tutorials in Optimization and Decision Support Technologies*. Springer.

Coello Coello, C., Lamont, G., and Van Veldhuizen, D. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, 2nd edition.

Coello Coello, C. A. and Cruz Cortés, N. (2005). Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, 6(2):163–190.

Czyzak, P. and Jaszkiewicz, A. (1998). Pareto simulated annealing - a meta-heuristic technique for multiobjective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1):34–37.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Chichester.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm - NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deb, K., Mohan, M., and Mishra, S. (2005). Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. *Evolutionary Computation*, 13(4):501–525.

Ehrgott, M. (2005). *Multicriteria Optimization*. Springer, 2nd edition.

Ehrgott, M. and Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spectrum*, 22(14):425–460.

Gandibleux, X., Sevaux, M., Sörensen, K., and T'kindt, V., editors (2004). *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*. Springer.

Garcia-Martinez, C., Cordon, O., and Herrera, F. (2007). A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria tsp. *European Journal of Operational Research*, 180(1):116–148.

Glover, F. and Kochenberger, G., editors (2003). *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston.

Grosan, C. (2004). Improving the performance of evolutionary algorithms for the multiobjective 0/1 knapsack problem using ϵ-dominance. In *Proc. of the 2004 Congress on Evolutionary Computation (CEC 2004)*, pages 1958–1963, Portland, USA. IEEE Press.

Hart, W., Krasnogor, N., and Smith, J., editors (2004). *Recent Advances in Memetic Algorithms*. Springer.

Ishibuchi, H., Yoshida, T., and Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223.

Jaszkiewicz, A. (2002). On the performance of multiple-objective genetic local search on the 0-1 knapsack problem - a comparative experiment. *IEEE Transactions on Evolutionary Computation*, 6(4):402–412.

Knowles, J. and Corne, D. (2000a). Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172.

Knowles, J. and Corne, D. (2000b). A comparison of diverse approaches to memetic multiobjective combinatorial optimization. In *Proc. of the Workshop on Memetic Algorithms (WOMA) at the 2002 Genetic and Evolutionary Computation Conference (GECCO 2000)*, pages 103–108.

Knowles, J. and Corne, D. (2004). *Memetic Algorithms for Multiobjective Optimization: Issues, Methods and Prospects*, volume 166 of *Studies in Fuzziness and Soft Computing*, pages 313–352. Springer.

Krasnogor, N. (2002). *Studies on the Theory and Design Space of Memetic Algorithms*. PhD thesis, University of the West of England.

Landa-Silva, J., Burke, E., and Petrovic, S. (2004). *An Introduction to Multiobjective Metaheuristics for Scheduling and Timetabling*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 91–129. Springer.

Larranaga, P., Kuijpers, C., Murga, R., Inza, I., and Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13:129–170.

Li, H. and Landa-Silva, D. (2008). Evolutionary multi-objective simulated annealing with adaptive and competitive search direction. In *Proc. of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008)*, pages 3310–3317, Hong Kong. IEEE Press.

Li, H., Landa-silva, D., and Gandibleux, X. (2010). Evolutionary multi-objective optimization algorithms with probabilistic representation based on pheromone trails. In *Proc. of the 2010 Congress on Evolutionary Computation (CEC 2010)*, pages 2307–2314, Barcelona, Spain.

Li, H. and Zhang, Q. (2009). Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302.

Merz, P. (2000). *Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen.

Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial Optimization : Algorithms and Complexity*. Dover.

Paquete, L. and Stützle, T. (2003). A two-phase local search for the biobjective traveling salesman problem. In *Proc. of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, pages 479–493. Springer-Verlag.

Sanghamitra, B., Sriparna, S., Ujjwal, M., and Kalyanmoy, D. (2008). A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation*, 12(3):269–283.

Serafini, P. (1992). Simulated annealing for multiobjective optimization problems. In *Procceedings of the 10th International Conference on Multiple Criteria Decision Making*, pages 87–96.

Smith, K., Everson, R., and Firldsend, J. (2004). Dominance measures for multi-objective simulated annealing. In *Proc. of the 2004 IEEE Congress on Evolutionary Computation (CEC 2004)*, pages 23–30, Portland, USA. IEEE Press.

Smith, K. I., Everson, R. M., Fieldsend, J. E., Murphy, C., and Misra, R. (2008). Dominance-based multiobjective simulated annealing. *IEEE Transactions on Evolutionary Computation*, 12(3):323–342.

Suman, B. and Kumar, P. (2006). A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society*, 57(10):1143–1160.

Tan, K., Khor, E., and Lee, T. (2005). *Multiobjective Evolutionary Algorithms and Applications*. Advanced Information and Knowledge Processing. Springer.

Tekinalp, O. and Karsli, G. (2007). A new multiobjective simulated annealing algorithm. *Journal of Global Optimization*, 39(1):49–77.

Ulungu, E., Teghem, J., Fortemps, P., and Tuyttens, D. (1999). MOSA method - a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236.

Zhang, Q. and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731.

Zitzler, E., Laumanns, M., and Thiele, L. (2002). SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Proc. of Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems (EUROGEN 2001)*, pages 95–100.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.